­

**Kick-Starter Kit for Big Data Developers**

[**Reference Links** 3](#_Toc514336059)

[**Hadoop** 3](#_Toc514336060)

[**Spark** 3](#_Toc514336061)

[**Scala** 3](#_Toc514336062)

[**Hive** 3](#_Toc514336063)

[**Kafka** 3](#_Toc514336063)

[**NiFi** 3](#_Toc514336063)

[**Test Datasets** 3](#_Toc514336064)

[**Onboarding Doc** 3](#_Toc514336065)

[**Prerequisites** 3](#_Toc514336066)

[**Beginner Guide** 4](#_Toc514336067)

[**Assignment 1** 4](#_Toc514336068)

[**Assignment 2** 5](#_Toc514336069)

[**Assignment 3** 6](#_Toc514336070)

[**Assignment 4** 7](#_Toc514336071)

[**Assignment 5** 8](#_Toc514336072)

[**Assignment 6** 8](#_Toc514336072)

[**Assignment 7** 8](#_Toc514336072)

[**Assignment 8** 8](#_Toc514336072)

[**Advanced Guide** 9](#_Toc514336073)

[**Assignment** 9](#_Toc514336074)

[**Spark Job Error Resolutions** 10](#_Toc514336075)

[**Spark Job Tuning** 10](#_Toc514336076)

[**Spark Job Tuning 2** 10](#_Toc514336077)

[**Spark GC Tuning** 10](#_Toc514336078)

[**Expert Guide** 10](#_Toc514336079)

# @Author : Atul / Naresh

# **Reference Links**

## **Hadoop**

* <https://hortonworks.com/apache/hadoop/>

## **Spark**

* <https://docs.databricks.com/spark/latest/gentle-introduction/gentle-intro.html#gentle-introduction-to-apache-spark>
* <https://databricks.com/blog/2016/07/14/a-tale-of-three-apache-spark-apis-rdds-dataframes-and-datasets.html>
* <https://www.youtube.com/watch?v=7ooZ4S7Ay6Y>

## **Scala**

* <https://databricks.com/blog/2016/08/15/how-to-use-sparksession-in-apache-spark-2-0.html>
* <https://docs.databricks.com/spark/latest/dataframes-datasets/introduction-to-dataframes-scala.html>

## **Hive**

* <https://www.facebook.com/notes/facebook-engineering/hive-a-petabyte-scale-data-warehouse-using-hadoop/89508453919/>

## **Kafka**

* [**https://kafka.apache.org/documentation/**](https://kafka.apache.org/documentation/)

## **NiFi**

* [**https://nifi.apache.org/docs.html**](https://nifi.apache.org/docs.html)

## **Test Datasets**

* This link will be used to download datasets to cover the assignments <https://catalog.data.gov/dataset>

# **Onboarding Doc**

* Share point link : <https://jiosp.ril.com/sites/BDCOE/Shared%20Documents/Onboarding/TeamOnboardingDoc.docx?Web=1>

# **Prerequisites**

1. Setup an IDE with scala packages added (team uses Intellij)
2. Install portable version of Mobaxterm (basically, follow the aforementioned onboarding doc)
3. Raise request to get access for RTC and share point
4. Get acquainted with basic linux Commands: ssh, ls -latr, du -sh, df, tail -f, ps –aux, vi, cd, cp, mv, mkdir –p, grep, wc , top, less
5. Go through the above reference links
6. Google the doubt first, before asking for help

# **Beginner Guide**

W222

## **Assignment 1**

## For below activities, use reference links given in the document and explore in internet.

* Hadoop Federation,Architecture and overview
* Yarn Architecture and overview
* Kafka Architecture, overview and different offset management strategies
* NiFi Architecture and overview
* Hive Architecture and overview
* Sqoop Overview
* Spark Overview and internal working of Spark Submit execution
* Kerberos Architecture

## **Assignment 2**

(Make sure all prerequisites are met)

1. Read a csv file into a data frame using scala
2. Clean the dataframe (rename the columns and remove special characters)
3. Save dataframe into a hive table
4. Query the hive table for counting the rows(using sparakSql api)

Read on :

1. What is Big Data?
2. What is spark ?
3. What is Hadoop ?
4. What is hive ?
5. Difference between spark context and spark session
6. Difference between RDDS, Dataframe, Dataset, SparkSql APIs

## **Assignment 3**

1. Read a csv file into a data frame using scala
2. Clean the dataframe (rename the columns and remove special characters)
3. Partition, bucket and sort the dataframe (decide and justify the columns used for partitioning, bucketing and sorting; and the number of buckets used)
4. Save the partitioned and bucketed dataframe into a hive table
5. Find the table description from Hive and check parameters of buckets, partitions, location for the newly ingested hive table.
6. Query the hive table and find how many partitions are formed

Read on :

1. Difference between Partitioning and bucketing
2. When to use partitioning, bucketing and sorting (pros and cons of each)
3. How to decide the optimal number of buckets ?
4. Difference between pruning and predicate pushdown and how does partitioning and bucketing leverage each ?
5. How to decide which columns to select for all the above three mentioned use cases (are there any data type preference for the columns ?)

* <https://community.hortonworks.com/questions/23103/hive-deciding-the-number-of-buckets.html>
* <https://qubole.zendesk.com/hc/en-us/articles/214885263-HIVE-Dynamic-Partitioning-tips>
* <https://data-flair.training/blogs/bucket-map-join/>

6)What are bloom filters in hive table?

## **Assignment 4**

1. Read a csv file into a data frame using scala (and find the count using dataframe api)

df.count()

1. Clean the dataframe (remove special characters from column names, if present)

Charset option while reading using spark.read.csv

1. Save the dataframe into hdfs path : "/tmp/spark\_assignment\_<yourname>/" in parquet format

df.write.parquet(<path>) or df.write.format(“csv/orc”).save(<path>)

1. Make another df by reading the above hdfs path

Spark.read.parquet/csv/text(<path>)

1. Find number of records in the data frame (using dataframe api)

Dff.count()

1. Store the dataframe as a view and find number of records (using sparkSql api)

Dff.createOrReplaceTempView(<name>)

Spark.sql(“Select count(\*) from <name>”)

1. Make an external table on hive and find the number of records (using beeline client)

Dff.write.option(“path”, <path>).saveAsTable(<name>)

8. Now find number of records of the newly created hive external table (using spark sql api)

9. Make sure all the counts in 1,5,6,7,8 match : what is difference between execution strategies of all 5 counts

Read on :

1. How and where is data sitting in a dataframe stored ?
2. How and where is data sitting in a hive table stored ?
3. What is the optimal format to store data in hive and its pros and cons ?
4. What is optimal format to store data in hadoop (while using spark to query the data) and its pros and cons ?
5. What is hive metastore ?
6. Difference between external and internal hive table, pros and cons, which scenarios are they used in?
7. What is Map Reduce?
8. Read on hive execution strategy
9. Difference in execution strategies of : Hive with MR mode, Hive on Tez, Hive with LLAP (which execution engine is used in every use case ?, pros and cons of each)

## **Assignment 5**

Download a gzip file and transfer it to dev server in local file system

1. Load the gzip file into hdfs (hadoop file system) from local file system (what is the best way to do it ?)

Hdfs must store entire file on a given datanode since gzip files aren’t splittable. But the replication factor would still be 3. A single mapper would have to process the entire file. On the contrary, formats such as bzip2 are splittable. Preferred due to parallel MR jobs.

<http://www.hadooplessons.info/2017/07/loading-compressed-data-into-hive-table.html>

1. Use spark scala to gunzip the file

Reading a zipped file in spark is done in same way as reading a normal file. If the file has tab delimiter then: spark.read.option(“sep”, “\t”).csv(<path>)

1. Identify the type of data format of the unzipped files and accordingly store it in a dataframe

<ASK>

1. Control the number of files to be written in hive table while storing the dataframe in hive (partitioned, bucketed, sorted) (in required format)

Repartition: increase or decrease # of partitions

Coalesce: optimized version of repartition for only decrease partitions

Can specify partitions when creating rdd but not df in spark. Nice.

1. Query the hive data

hiveContext.sql or spark.sql don’t forget to enable hive support when creating spark.

1. Find the difference in size and number of input files(files created after unzipping the gzip file) and output files(in the hive table) [Hint: Use shell commands]

<ASK>. Why the size difference?

1. Justify the above differences

Read on :

1. How to control the number of files created while saving a dataframe into a partitioned and bucketed hive table ?
2. What are partitions in a dataframe
3. What is difference between repartitioning and coalesce, pros and cons, how to decide which one to use ?
4. Understanding how a spark job execution works ?
5. What is a driver and an executor
6. What is a job, stage, dag, task in spark parlance and what factors determine them ?

## **Assignment 6**

1. Download two datasets
2. Store both the downloaded files in hdfs
3. Read both the data in different dfs (have proper names for every df)
4. Add a column in both the dfs named rank and populate it with the row number of every row
5. cache one dataframe, and persist the other (using dataframe apis)
6. Also create temp views for both the dfs
7. Create the following dataframes : (all the below joins will be on the newly create column : rank)

* df\_left = df1 left join df2 (using data frame api) (show 10 rows of this dataframe)
* df1.join(df2, $"df1Key" === $"df2Key", "left”)
* df\_left\_sql = tempview1 left join tempview2 (using sparkSql api) (show 10 rows of this dataframe, and validate with above)

create 2 tempview as df.createorReplaceTempView() and spark.sql(“Select col\_names(s) from view1 LEFT JOIN view2 ON col\_name==col\_name”)

* df\_right = df1 right join df2 (using data frame api) (sort by rank, and show 5 rows)
* df\_right\_sql = tempview1 right join tempview2 (using sparkSql api) (sort by rank, and show 5 rows, and validate with above)
* df\_inner = df1 inner join df2 (using data frame api) (use collect on this dataframe)
* df\_inner\_sql = tempview1 inner join tempview2 (using sparkSql api)
* df\_outer = df1 full outer join df2 (using data frame api) (use collect on this dataframe)
* df\_outer\_sql = tempview1 full outer join tempview2 (using sparkSql api)

Read on :

1. Different types of sql joins (inner, left, right, full outer)
2. Difference between cache and persist (how many types of persistence are possible), pros cons, in which scenarios to use each.
3. When is a variable broadcast used ?
4. What does collect on a dataframe do ?
5. Difference between join mechanisms available in spark : Broadcast Hash join , Shuffle hash join , Sort merge join

1. Difference between Hive bucketing support and spark bucketing support <https://databricks.com/session/hive-bucketing-in-apache-spark>

## **Assignment 7**

* Coordinate with [Pawan1.Chauhan@ril.com](mailto:Pawan1.Chauhan@ril.com)/ [Sreedhar.Rapolu@ril.com](file:///C:\Users\mamta.nainwal\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\4KBF3LNB\Sreedhar.Rapolu@ril.com) and understand end to end Sample NiFi Flow used and try doing the same by your own.
* Sample assignment would be given directly by Pawan/Sreedhar

## **Assignment 8**

 

1. Load employee and department dataset into 2 different table :
   1. Emp\_orc
   2. Dep\_orc
2. Create a kpi table(emp\_kpi) having department wise 2nd highest salary of an employee. Below are the sudo logic for that :
   1. Join employee and department table
   2. Find 2nd highest salary of each department
   3. Schema should be like :  
      emp\_id, emp\_name(concat fname and lname),emp\_age ,dep\_id,dep\_name
3. Write a spark job to write corresponding hive table(emp\_kpi) to es index(get/create the sample index name of dev cluster)

# **Advanced Guide**

## **Assignment**

* Dissect the below mentioned shell script and create a flow diagram for the same.
* Shell Script : hive\_compressor.sh -- Share point Link :-<https://jiosp.ril.com/sites/BDCOE/Shared%20Documents/hive/Custom%20Hive%20Compressor/hive_compressor.sh>
* HiveCompressorScript.docx  : All about the script's algorithm, configs, and works under development [https://jiosp.ril.com/sites/BDCOE/Shared%20Documents/hive/Custom%20Hive%20Compressor/HiveCompressorScript.docx?Web=1](https://jiosp.ril.com/sites/BDCOE/Shared%20Documents/hive/Custom%20Hive%20Compressor/HiveCompressorDoc.docx?Web=1) ( Share point Link )

Read on :

* How many types of hive compressions are there ?
* Difference between a major and minor compact in hive
* Difference between a transactional and non transactional table in any DBMS
* Difference between hive table compaction and concatenation
* How to manually perform compaction and concatenation on transactional(partitioned, and non-partitioned) and non-transactional(partitioned, and non-partitioned) hive tables
* Possible resolutions when hive compactions or concatenations fail ?
* What is a vertex in a hive query ? How is the number of vertices decided ?
* In what scenarios do we get vertex failures (and there resolutions)
* How to perform compression on bucketed hive tables ?
* How to compress data in a partitioned, external table ?

## **Spark Job Error Resolutions**

* <https://jiosp.ril.com/sites/BDCOE/Shared%20Documents/Spark/SparkTuning-Part1.docx?Web=1>

## **Spark Job Tuning**

* <https://jiosp.ril.com/sites/BDCOE/Shared%20Documents/Spark/SparkTuning-Part2.docx?Web=1>
* Resource Planning  
  <https://wiki.jio.com/download/attachments/29116897/Spark%20Resource%20Planning%20and%20Tuning.docx?api=v2>

## **Spark Job Tuning 2**

* <https://jiosp.ril.com/sites/BDCOE/Shared%20Documents/Spark/Tuning%20Spark%20Job.docx?Web=1>

## **Spark GC Tuning**

* <https://jiosp.ril.com/sites/BDCOE/Shared%20Documents/Spark/SparkGCTuning.docx?Web=1>

## **Zoomdata**

* https://wiki.jio.com/pages/viewpageattachments.action?pageId=29100869&sortBy=date&highlight=Zoomdata+Technical+Documentv2.doc&highlight=Zoomdata+Starter+Kit.docx&

# **Expert Guide**

Review the above and start contributing !